

MC68000アドレス空間の問題点と構成方式

武部 桂史、鵜澤 繁行、中川 正樹、阿刀田 央一、高橋 延匡

(東京農工大学 工学部 数理情報工学科)

1. はじめに

従来の8ビットマイクロコンピュータは、アドレス空間として、 $2^{16}=64$ キロバイト程度が主流であった。16ビットマイクロコンピュータの出現は、従来のマイクロプロセッサのイメージ（用途としての簡単なコントローラ、簡単なパーソナルコンピュータ）を大幅に塗り替えるものであった。すなわち、第一に、基本的処理能力の向上（公称2 MIPS）、第二に、アドレス空間の大幅な拡張がある。たとえば、MC68000を例にとれば、 $2^{24}=16$ メガバイトのアドレス空間を持っている。これは、IBM360、IBM370の基本的アーキテクチャが持っていたアドレス空間と一致する。第三として、汎用大型計算機に見られるスーパーバイザ領域とユーザ領域の分離、それぞれの領域におけるデータとプログラムの分離の思想などが盛り込まれている。これは、メモリマネージメントユニット（あるいは、ダイナミックアドレストランスレータ）^{*}の存在が前提となっている。

本論文では、MC68000アーキテクチャの特徴であるスーパーバイザ用領域とユーザ用領域の、独立した2種類のアドレス空間を有効に生かすためのオペレーティングシステムとアーキテクチャとの関連について考察し、あわせて、我々が考案したアドレス空間構成の方式について報告する。

2. アーキテクチャとアドレッシング方式

MC68000は、HMOS技術を用いた16ビットマイクロコンピュータである。

主な特徴は次のとおりである。

- ・32ビットデータレジスタ(8個)とアドレスレジスタ(9個)(図2-1)
- ・16メガバイト直接アドレス空間
- ・ビット、BCD、バイト、語(16ビット)、長語(32ビット)の5種類のデータタイプ
- ・メモリマップドI/O
- ・14種のアドレッシングモード
- ・73種の命令
- ・動作周波数8メガヘルツ
- ・約2 MIPSの処理能力
- ・ユーザのデータ領域とプログラム領域、スーパーバイザのデータ領域とプログラム領域の4種類の独立したアドレス空間(16メガバイト $\times 4=64$ メガバイト)
- ・トレース機能
- ・MC6800の周辺デバイス群を使用可能
- ・例外ベクトル表を有し、ベクトル割込みが可能(例外ベクトル表は、スーパーバイザ領域の0~1023番地に存在する)

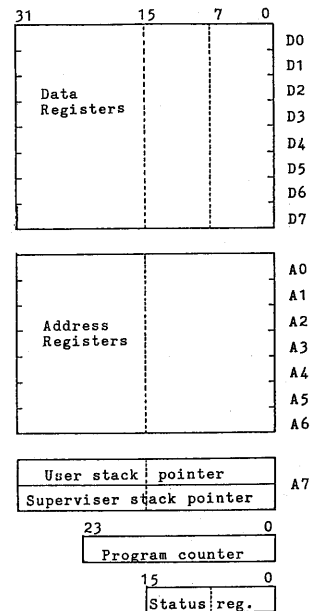


図2-1 MC68000 レジスタ構成図

このように、MC68000は中型コンピュータに近いアーキテクチャを有している。

本論文で問題としているスーパーバイザのデータ領域とプログラム領域、ユーザのデータ領域とプログラム領域の分離は、次のようにして行なわれる。

CPUには、スーパーバイザとユーザという2通りの内部状態が存在する。そして、その状態がどちらであるか、および現在のメモリ参照がプログラムかデータかによって、表2-1のような値をFC0-FC2（ファンクションコード）のピンに出力する。

*）ここで言うメモリマネージメントユニットは、仮想アドレスを実現するのに必要な機能を備えたものをさす。

もし、FC0-FC2のピンの出力をデコードすれば、同一のアドレス指定で、スーパーバイザプログラム、スーパーバイザデータ、ユーザプログラム、ユーザデータのそれぞれを、別々に参照することができる。言い換えれば、スーパーバイザプログラム、スーパーバイザデータ、ユーザプログラム、ユーザデータ各々に、独立した16メガバイトのアドレス空間を持たせることができる。もちろん、普通のマイクロコンピュータのように、アドレス空間をまったく分離しなくてもよいし、また一部（たとえば、スーパーバイザ、ユーザのように）分離してもよい。これは、MC68000チップを使用してシステムを設計する者の目的によって選択され得る。

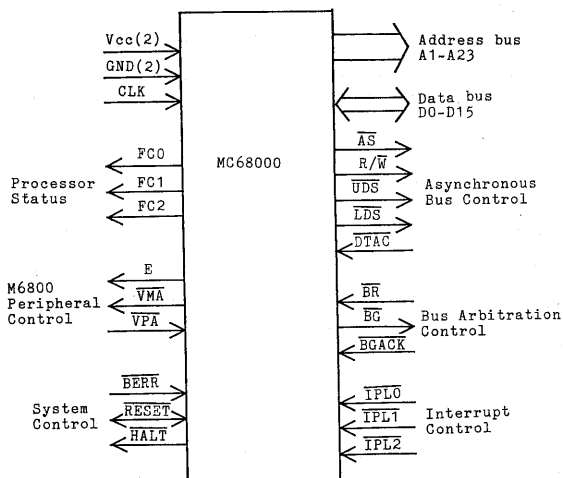


図2-2 MC68000 入出力信号

このような特徴を生かして、スーパーバイザ領域とユーザ領域を分離すると次のような利点が得られる。

- (1) ユーザとスーパーバイザがそれぞれ独立したアドレス空間を持つために、相互間の読み書きはできなくなる。すなわち、ほとんど自動的にメモリプロテクションの一機能が形成される。
- (2) ユーザのアドレス空間は、ユーザが自由に使える。
- (3) 実メモリシステムにおいて、オペレーティングシステムが拡張されても、ユーザのアドレス空間は変わらない。

MC68000のアーキテクチャの特徴をうまく発揮することは、アドレス空間をいかにうまくインプリメントするかにかかっている。

3. アドレス空間構成上の問題点

3.1 問題提起

MC68000についての様々な特徴、利点を述べてきたが、ここで、本論文が扱う問題をはっきりさせておく意味で、問題点を次に列記する。

- (1) スーパーバイザとユーザ間の交信をいかに行なうか。
- (2) スーパーバイザとユーザとの領域を分離した利点をいかに発揮させるか。

以後これらの問題について考察する。本報告では、データとプログラムを分離することは、説明を複雑にするので、スーパーバイザとユーザの領域を分離した場合について考える。

3.2 単一ユーザ向けシステムと複数ユーザ向けシステムとの対比

システムを単一ユーザ向けにするか、複数ユーザ向けにするかは、設計段階で大きく異なる。

(1) オペレーティングシステムについて

単一ユーザ向け	複数ユーザ向け
・システムプログラムはリロケータブルに作る	・システムプログラムはリロケータブルでリエントラントに作る
・入出力や表は単一ユーザ専用となる	・入出力や表の処理をシリアルリユーザブルに作るところがある

(2) プロテクション機構について

単一ユーザ向け	複数ユーザ向け
・オペレーティングシステムとユーザの間にだけプロテクトをする	・オペレーティングシステムと複数ユーザそれぞれの間にプロテクトをする

表2-1 ファンクションコード

参照領域

ファンクションコード	参照領域
FC2 FC1 FC0	
0 0 0	(未定義)
0 0 1	ユーザデータ
0 1 0	ユーザプログラム
0 1 1	(未定義)
1 0 0	(未定義)
1 0 1	スーパーバイザデータ
1 1 0	スーパーバイザプログラム
1 1 1	イントラプトアフレッジ

どちらの場合も、2キロ語程度ごとにメモリに錠をつけるか、外付けレジスタによる境界区分方式のメモリプロテクション機構あるいはそれに代わる機構が必要である。

(3) 入出力やオペレーティングシステムの表管理について

単一ユーザ向け	複数ユーザ向け
・入出力や表が専用化される	・1時期に1ユーザのためだけに入出力や表があるようにシリアルリユーザブルでなければならない

複数ユーザ向けシステムであれば、当然単一ユーザでも使用できる。以下、本報告では、複数ユーザを前提としてオペレーティングシステムを設計した場合について、アドレス空間に関する考察を行なう。

3.3 スーパーバイザとユーザ間の交信の必要性和そのデータ量

スーパーバイザとユーザ間の交信は、次のような場合が考えられる。ただし、ここでは、スーパーバイザ領域とユーザ領域は完全に分離し、かつ、スーパーバイザプログラムおよびスーパーバイザ用データはスーパーバイザ領域に存在し、ユーザプログラムとユーザ用データはユーザ領域に存在すると仮定する。

(1) ユーザプログラムの進行にともない、ユーザプログラムからスーパーバイザプログラムに対し、次のような各種の要求がだされる場合。

- 入出力関係（入出力の要求）
- プロセス管理関係（プロセスの起動、停止、同期）
- 主メモリ管理関係（主メモリの要求）
- その他

すなわち、スーパーバイザコールによるユーザプログラムとスーパーバイザプログラム間の交信が必要となる。

これらの場合には、パラメータの受け渡しに、4～7個のレジスタが使用できれば、十分対処可能である。

(2) スーパーバイザコールの中で、スーパーバイザの中の管理表を書き換えるだけでなく、実際に多量のデータ自身を移動させる場合。

- 同期をとって、1つのタスクから他のタスクにデータを渡す場合
- 2次メモリとユーザ領域間でロード、ストアを行なう場合
- コンパイラ、エディタ等がユーザの多量なデータを扱う場合
- その他

スーパーバイザが、スーパーバイザ領域にあると仮定すると、これらの交信を実行する場合には、数キロ語の共有領域が必要となる。

(3) 従来の単一アドレス空間では問題にならなかった、スーパーバイザとユーザ間の交信。

- unimplemented instruction TRAP
- illegal instruction 機能
- トレース機能（メモリダンプ）
- その他

これらは、スーパーバイザ領域からユーザ領域をアクセスできないということに、起因している。たとえば、unimplemented instruction TRAPなどでは、トラップ発生時に、スーパーバイザスタックにプログラムカウンタとステータスレジスタが退避されるが、スーパーバイザがこれらの値を用いて命令を解析しようとしても、スーパーバイザはユーザ領域を見ることができないため処理できない（図3.3-1）。

以上、MC68000のアーキテクチャでは、スーパーバイザ領域、ユーザ領域の区別はあっても、このままではオペレーティングシステムはそれを利用できないことが判明した。次の4章では、上記の解決策について論ずる。

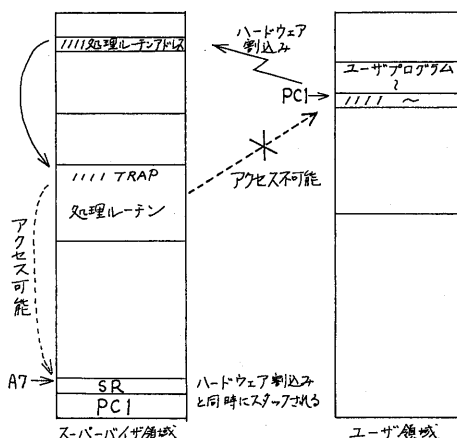


図3.3-1 unimplemented instruction TRAPの処理

4. アドレス空間の構成案

ここでMC68000のアドレス空間をメモリマネージメントユニットを用いなく（仮想メモリ方式とせずに）、インプリメントする構成を考える。

また、アドレス空間を考える上で、次の3点に重きを置いた。

- (1) 大きなアドレス空間を張る
- (2) スーパーバイザ領域とユーザ領域を分離する
- (3) それぞれの領域において、プログラム領域とデータ領域とを分離可能とする

4.1 レジスタを交信領域にする構成

数個の専用レジスタを設け、スーパーバイザ領域とユーザ領域の領域間の情報は、これらのレジスタを介してお互いに交信するシステムを考えてみる（図4.1-1）。

本構成の特徴は、MC68000のアーキテクチャで許されたアドレス空間を完全にインプリメントできる点にある。すなわち、スーパー

バイザとユーザのそれぞれの領域に16メガバイトのアドレス空間をインプリメント可能に設計するものである。

本構成にオペレーティングシステムをインプリメントする場合には、ジョブの制御をするルーチン、コンパイラ、エディタ等は、ユーザ領域に入れて使うことになろう。しかし、これらを完全に常駐させることは、スーパーバイザ領域とユーザ領域を分離させた利点を無駄にしてしまう。すなわち、ユーザ領域にはロードが常駐し、ブートストラップ方式によって、レジスタを介して、種々な処理ルーチンを、ユーザ領域に移す。完了後、スーパーバイザが、ユーザ領域で種々な処理を行なう。

本構成の利点は次のとおりである。

- (1) MC68000で用意されたアドレス空間をほぼ完全に使える。
- (2) ユーザとスーパーバイザ間のプロテクションは一応持てる。
- (3) ハードウェアのインプリメントが容易である。

本構成の問題点は次のとおりである。

- (1) 交信領域がレジスタだけのため、転送レートが小さい。
- (2) ユーザ領域に、オペレーティングシステムの一部が常駐する。
- (3) スーパーバイザ領域から、ユーザ領域のデータをアクセスできないため、これらのデータを必要とするプログラムは、ユーザ領域に入れなければならない。

このような構成では事実上、スーパーバイザ領域に常駐する部分のごく限られたものとなろう。

4.2 共有領域を交信領域にする構成

本構成は、4.1の構成において、交信データが多量になると性能が低下することを考え、スーパーバイザ領域とユーザ領域の間に共有領域を設けるものである（図4.2-1(i)）。すなわち、共有領域にはファンクションコー

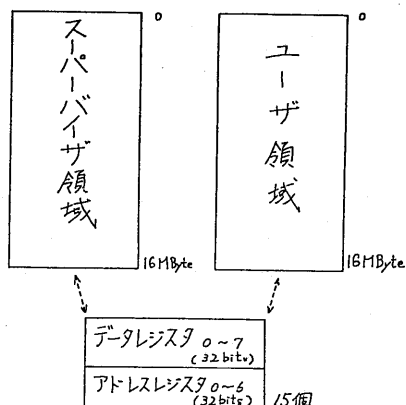


図4.1-1 レジスタを交信領域にする構成

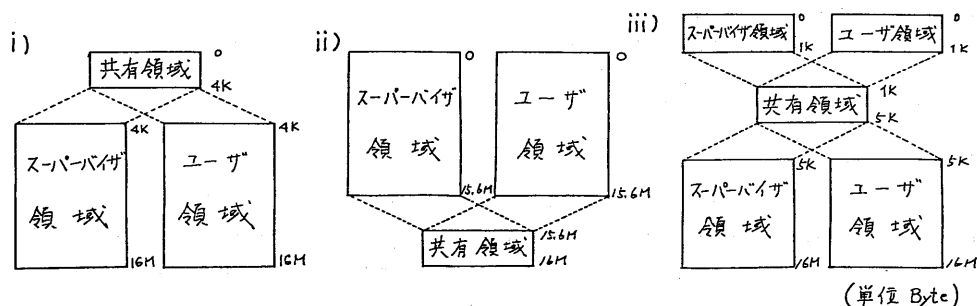


図4.2-1 共有領域を交信領域にもつ典型的な構成例

ド (FC0-FC2) のデコード出力を重ねて、スーパーバイザからもユーザからもアクセス可能にする。

本構成の特徴は、多量のデータを必要とする交信でも、性能を低下させずに実行できることにある。

本構成の利点は次のとおりである。

- (1) 交信領域が大きいため情報のやり取りが容易である。
- (2) ジョブ間のプロテクション機構を除いてハードウェアのインプリメントが容易である。
- (3) ユーザ領域にオペレーティングシステムが常駐しなくてよい。

本構成の問題点は次のとおりである。

- (1) ユーザ領域から、共有領域の部分が削られる。
- (2) ユーザ領域にマルチプログラムで複数のユーザのプログラムが入っているとすると、共有領域はユーザ数だけ分割する必要がある。この場合、共有領域をユーザごとにプロテクトすることは、インプリメントの面から難しい。
- (3) ユーザ領域と共有領域の間でプロテクション機構が必要である。

共有領域を設けたモデルとして、3方式が考えられる (図4.2-1)。(i)の場合は、共有メモリ内に例外ベクトル表が存在する。これは、ユーザ領域からプロテクトしなければならず、好ましくない。(ii)の場合は、アドレス空間の最後に共有領域があるため、ユーザはアドレス空間をスーパーバイザに占有されたという感じを抱かせなくてすむ。(iii)の場合は、(i)の場合において、ハードウェアの付加を少なくするために、例外ベクトル表の部分を分割したものである。

4.3 スーパーバイザ領域とユーザー領域を重ねた構成

本構成は、従来のアドレス空間の考え方と同様に、単一アドレス空間(16メガバイト)を持ち、ここに、スーパーバイザ領域とユーザ領域を設ける構成である。しかし、この場合のユーザ領域は、スーパーバイザ領域でもあるため、従来の単一アドレス空間のアーキテクチャと同様にシステムをインプリメントすることが可能である。また、ユーザはスーパーバイザ常駐部にはアクセスできないようにする。

本構成の利点は次のとおりである。

- (1) アドレス空間のすべてがスーパーバイザ領域のため、交信領域を特別に持つ必要がない。
- (2) ユーザからスーパーバイザを容易に保護できる。
- (3) 従来のコンピュータシステムの構成法の中心をなすものであり、ソフトウェアが作り易い。

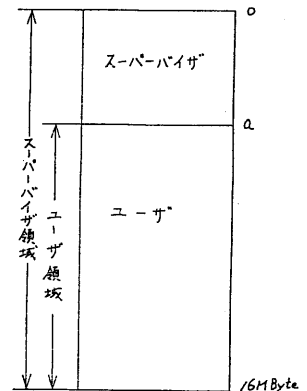


図4.3-1 スーパーバイザ領域とユーザ領域を重ねた構成

本構成の問題点は次のとおりである。

- (1) アドレス空間として、ユーザ領域とスーパーバイザ領域の合計で16メガバイトしか実装できない。
- (2) オペレーティングシステムを拡張すると、ユーザ領域が削られる可能性がある。

4.4 簡易型メモリマネジメントユニットを使用する構成

本構成は、スーパーバイザ領域とユーザ領域は、それぞれ16メガバイトのアドレス空間を張るが、物理メモリをブロック化して、そのブロック単位でダイナミックに属性、アドレスを変えられるようにしようとするものである。

すなわち、各メモリブロックに、プログラム

でアクセス可能なレジスタを設け、このレジス

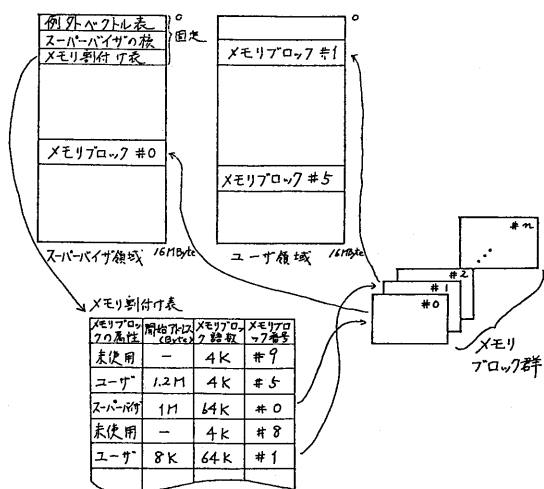


図4.4-1 簡易型メモリマネジメントユニットを使用する構成

タと実際のファンクションコード、アドレスとを比較して、各メモリブロックの属性とアドレスを決定するものである(図4.4-1)。

この場合のメモリブロックは、4キロ語単位の固定長も考えられるし、可変長も考えられる。もし、固定長であれば、メモリボード単位ごとに、このようなレジスタを持てばよいことになる。

この構成で、例外ベクトル表、スーパーバイザの核、メモリ割付け表だけは、固定にしておくことが望まれる。

本構成の利点は次のとおりである。

- (1) アドレス空間として、スーパーバイザとユーザの領域が完全に分離できる。
- (2) メモリブロックの属性、アドレスが自由に変えられる。
- (3) 全アドレス空間分のメモリがなくても、アドレスは自由に使える。

本構成の問題点は次のとおりである。

- (1) ハードウェア、ソフトウェアともに複雑になる。そのためにシステムの信頼性が落ちる危険性がある。
- (2) メモリブロックの大きさが決まっているため扱いにくい。

4.5 出窓式簡易メモリ管理ユニットを使用する構成

4.4の構成では、全メモリブロックの属性、アドレスを、ダイナミックに変えることができるため、ハードウェア、ソフトウェアともに複雑であった。たとえば、属性、アドレスをダイナミックに変えるためには、常時全メモリブロックの内容をスーパーバイザが管理しなければならないし、ハードウェアとしても、メモリアクセスごとにレジスタとの比較を行なうため、高速のコンパレータを持つことが望まれる。また、アドレスを自由に変更できることは、オペレーティングシステムのオーバーヘッドが多くなる。そこで、固定領域を設けて、そこをスーパーバイザとユーザ間の共有領域として使う構成を考えた。

本構成は、スーパーバイザの固定領域(出窓)に、ユーザ領域のあらゆるメモリブロックを割り当てられるようにした構成である。割り当てられたメモリブロックは、スーパーバイザのデータとして扱われる。

本構成の利点は次のとおりである。

- (1) 4.4の構成より、ハードウェア、ソフトウェアともインプリメントが容易である。
- (2) スーパーバイザが全ユーザ領域にアクセスできる。
- (3) アドレス空間として、スーパーバイザとユーザの領域が完全に分離できる。

本構成の問題点は次のとおりである。

- (1) メモリブロックの大きさが決まっているため扱いにくい。
- (2) 必要な情報が、複数のメモリブロックに存在する場合、固定領域では性能が落ちる。

4.6 共有領域の属性をダイナミックに変える構成

4.2の構成は、共有領域の保護に問題があった。4.5の構成は、共有領域の保護は行なえるが、ハードウェアが4.4ほどではないにしても複雑である。そこで本構成では、4.2の構成と4.5の構成を取り入れたものである。すなわち、共有領域を、スーパーバイザ領域にするか、ユーザ領域にするかを、ダイナミックに変更させるものである(図4.6-1)。

本構成では、4.2、4.4、4.5の構成の特徴を有している。

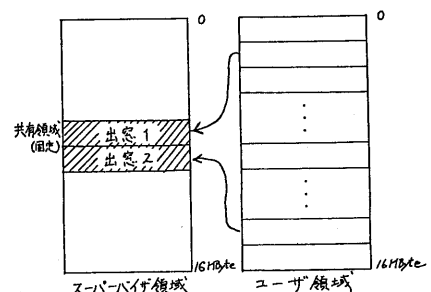


図4.5-1 出窓式簡易メモリ管理ユニットを使用する構成

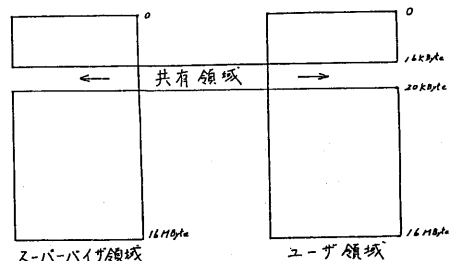


図4.6-1 共有領域の属性をダイナミックに変える構成

4.7 スーパーバイザの参照するデータをフラグにより切り換える構成

4.4、4.5、4.6の構成では、メモリブロック毎に属性を変更するものであった。しかし、スーパーバイザとユーザ間の交信を行なう上では、スーパーバイザのデータは、スーパーバイザ領域にあるか、ユーザ領域にあれば良い。スーパーバイザの参照するデータをフラグにより切り換える構成では、プログラムで変更可能なフラグ（以後、データ参照スイッチと呼ぶ）を設け、これと実際のファンクションコードとを比較して、スーパーバイザのデータに対して擬似ファンクションコードを作り出すものである（図4.7-1）。

データ参照スイッチの役割は、プログラムカウンタに関係する制御を変更するものではなく、データとして参照する領域を切り換えるものである。この構成を用いると、コンパイラのようなプログラムをスーパーバイザ領域に割り付けて、ユーザ領域のソースプログラムをデータとして処理することができる。

本構成において、例外ベクトル表、スーパーバイザスタック、I/O領域の部分のアドレスは、ユーザ領域から削除している（理由は次の章で述べる）。また、ユーザ領域からこれらの領域（0～16キロバイト）はアクセスできないようにインプリメントされている。

本構成の利点は次のとおりである。

- (1) スーパーバイザとユーザ領域の2つの属性を持つ領域を設定する必要がない。
- (2) ユーザからスーパーバイザの保護ができる。
- (3) コンパイラ、エディタ等の、ユーザプログラムを処理するものであっても、スーパーバイザ領域に置くことができる。
- (4) スーパーバイザは、ユーザ領域をすべてアクセスできる。
- (5) スーパーバイザ領域とユーザ領域のそれぞれにおいて、データ領域とプログラム領域を分離させることができる。
- (6) ユーザ領域で開発したプログラムを、スーパーバイザ領域に移すことができる。
- (7) ハードウェアのインプリメントが容易である。

本構成の問題点は次のとおりである。

- (1) データ参照スイッチに関するハードウェアが必要になる。

- (2) ユーザ領域の0～16キロバイトが削除される。

我々は、本構成をMC68000のアドレス空間として採用したので、詳しくは次章で述べる。

5. “スーパーバイザの参照するデータをフラグにより切り換える構成”の選定理由

メモリマネージメントユニットを使わずに、MC68000のアドレス空間をいかに使うかという問題に対して、様々なモデルを考えた。

それぞれの利点、問題点を考え合わせて、4.7の構成を我々は採用することにした。採用理由としては次のとおりである。

- (1) ユーザ領域の0～16キロバイトを削除したが、16キロバイト～16メガバイトは、スーパーバイザ領域とユーザ領域とを分離できる。
- (2) 特殊なハードウェア機構は、ファンクションコードの修飾機構のみとなり、インプリメントが容易である。
- (3) スーパーバイザ領域から全ユーザ領域をアクセスできることにより、スーパーバイザ関係のプログラムをすべてスーパーバイザ領域に割り当てることができる。
- (4) 構成が単純なため、ハードウェア、ソフトウェアともに信頼性の向上が容易にできる。
- (5) 将来、領域をプログラム、データに関しても分離するような拡張に対しても、容易に対処できる。

スーパーバイザの参照するデータをフラグにより切り換える構成のインプリメントに関して、さらに詳しく述べる。

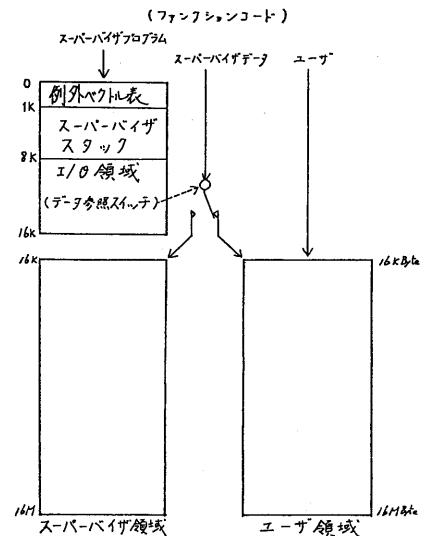


図4.7-1 スーパーバイザの参照するデータをフラグにより切り換える構成

MC68000では、メモリをアクセスする場合、アドレスとファンクションコードをデコードして行なう。本方式では、ファンクションコードを擬似的に作り出して、スーパーバイザ領域かユーザ領域かの選択を行なうものである。擬似ファンクションコードは、実際のファンクションコードと、プログラムでアクセス可能なレジスタ（データ参照スイッチ）によって自動的に作り出される。本構成では、スーパーバイザがデータをアクセスするとき、スーパーバイザ領域かユーザ領域かに選択できれば良いので、スーパーバイザのデータというファンクションコードが出力されたとき、データ参照スイッチと比較して、スーパーバイザのデータ、ユーザのデータ、スーパー

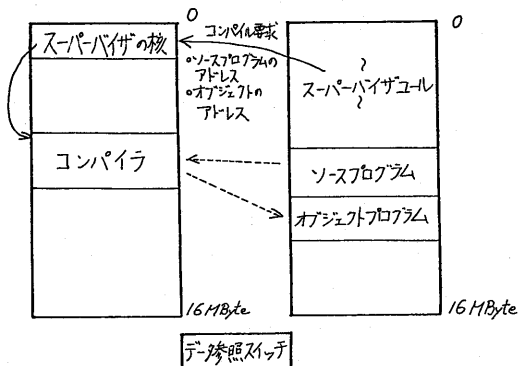


図5-1 データ参照スイッチの例

バイザのプログラムあるいはユーザのプログラムという擬似ファンクションコードを出力すればよい。この構成をとると、スーパーバイザは、その領域内から、ユーザの領域をすべてアクセスすることが可能となる。

たとえば、コンパイラを例にとってみよう。この場合、ユーザからソースプログラムをコンパイルする要求が出されたとする。スーパーバイザの核の部分でそれを認識して、ソースプログラムの存在するアドレスとオブジェクトプログラムを出力するアドレスをパラメータとしてコンパイラに渡し、制御を移す。コンパイラは、データ参照スイッチをユーザ領域に設定して、ソースプログラムをコンパイルする。

本方式ではメモリ領域を完全に分離し、システムプログラムの拡張にともなうユーザ領域の減少を避けることができる。

MC68000がI/Oポートを有し、データ参照スイッチをアドレス空間外に付けられれば、各領域のアドレス空間として0~16メガバイト得られる。しかし、現実には、次のような問題を有している。

- (1) MC68000はメモリマップドI/O方式を採用しているため、データ参照スイッチをアドレス空間に埋め込む必要がある。
- (2) データ参照スイッチは、スーパーバイザだけがアクセス可能でなければならない。
- (3) データ参照スイッチで、データ参照をユーザ領域に設定した後、次のデータ参照をスーパーバイザ領域にするために、データ参照スイッチをアクセスする必要がある。
- (4) データ参照スイッチが、ユーザのデータに設定されていた場合に、割込みが発生すると、スーパーバイザスタックポインタの指す α 番地に、プログラムカウンタ、ステータスレジスタを退避しようとするが、実際は、ユーザ領域の α 番地に、これらを退避してしまう（図5-2）。次に、スーパーバイザ領域の例外ベクトル表 β 番地をアクセスして、 γ 番地に飛んで割込み処理をしようとするが、実際には、ユーザ領域の β 番地をアクセスして、スーパーバイザ領域の δ 番地を割込み処理ルーチンとして制御を移してしまう。

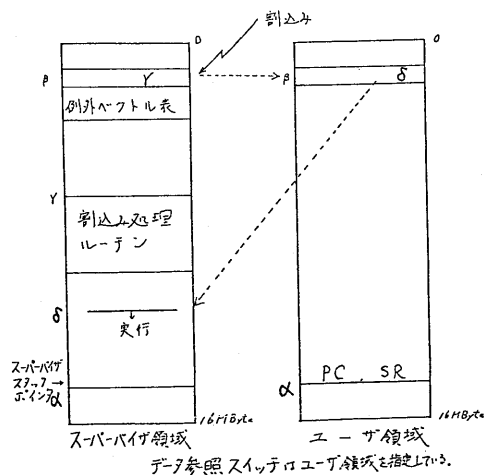


図5-2 割込み時の問題点

以上の問題の解決策として、次の二種類の方法が考えられる。

- (1) データ参照スイッチのための1アドレスを、ユーザ領域とスーパーバイザ領域に重ね合わせ、割込み発生時には、データ参照スイッチを無視するようにする。

(2) データ参照スイッチ、スーパーバイザスタック、例外ベクトル表の領域をユーザ領域から削る。

(1)の方式の最大の問題点は、アドレスエラーなどIC外部に信号が出力されない特殊な割込みに対処できないことと、ハードウェア的に複雑になることである。一方(2)の方式は、ハードウェアのインプリメントも容易であり、方式自身も単純であることから、我々は(2)の方式を採用し、図5-3のようなアドレス空間を張った。

データ参照スイッチの回路図は、大略図5-4の示すように、単純である。また、データ参照スイッチは、読み書きが可能になっている。

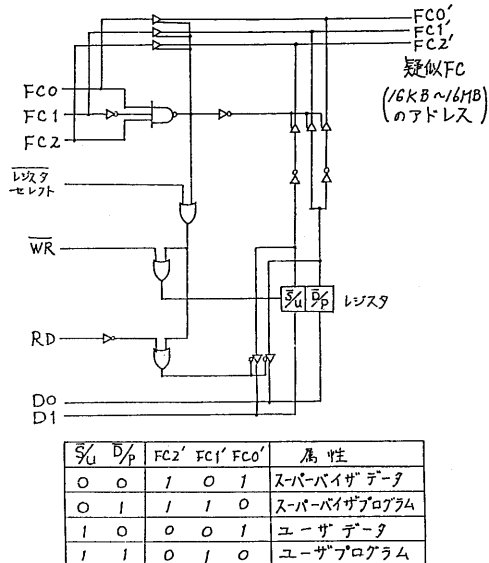


図5-4 データ参照スイッチ回路図

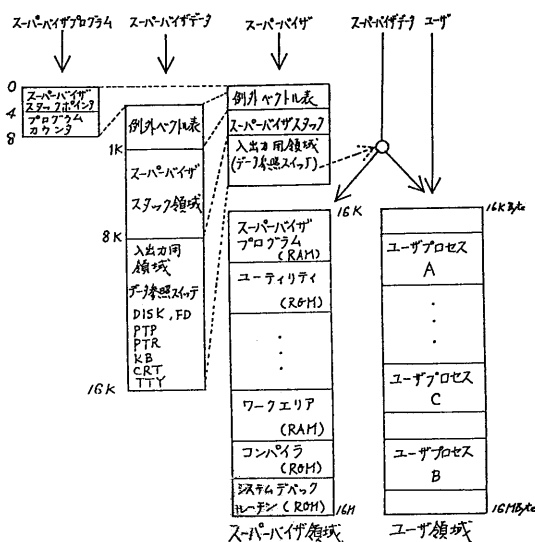


図5-3 採用したMC68000アドレス空間

6. 本構成でのシステム構成

MC68000は、ミニコンピュータから中型計算機なみのアーキテクチャを有しているマイクロコンピュータなので(最小構成としても)、入出力機器として、紙テープリーダ、紙テープパンチ、キーボード、キャラクタディスプレイ、ディスクなどが接続されると考えられる。メモリに関しても、65キロRAMなど、高集積化とともに、ビット当りの価格が大幅に低下傾向を示しており、多量のメモリを実装することが可能となる。ROMの値段の低下は、従来ディスクなどに格納されていたシステムプログラムを、ROM化して常駐させることも考えられるであろう。

現在、ここで提案したアドレス空間構成を有する、

図6-1の構成を持つMC68000シングルプロセッサシステムを作製中であり、一部作動している。

システムは、電源投入と同時にMC68000にリセットをかける。MC68000は、リセットされるとスーパーバイザプログラム領域の0番地と4番地(長語)を、スーパーバイザスタックポインタ(A7)とプログラムカウンタ(PC)として取り込み、そのプログラムカウンタより実行する。

本システムでは、例外ベクトル表をスーパーバイザデータに割り当て、

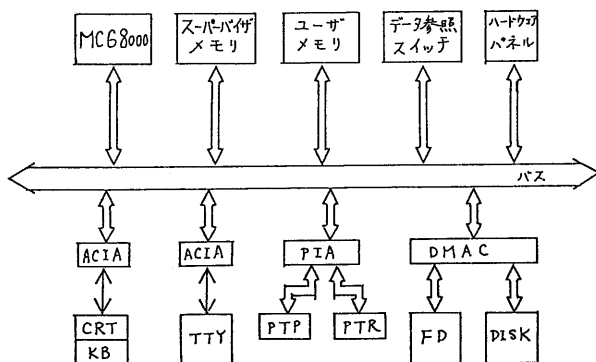


図6-1 MC68000シングルプロセッサシステム構成図

0番地と4番地(長語)の部分だけを、スイッチ化して、スーパーバイザプログラムに割り当てた。このことにより、電源投入と同時にシステムモニタなどを、実行させることができる。

7. まとめ

6章で述べた方式を、MC68000シングルプロセッサシステムに採用した。

メモリの価格低下は、多量のメモリを実装することをますます実現的なものになっている。し

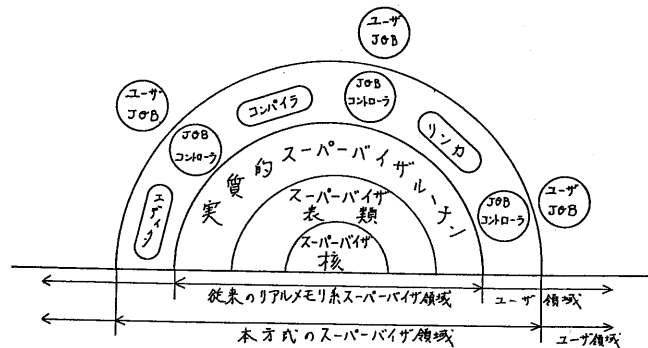


図7-1 オペレーティングシステム概念図

たがって、その動向をシステムの拡張に生かせる設計がなされるべきである。そのことはシステムのライフタイムを長くすることにも役立つ。

本方式は、アドレス空間を拡張できるばかりでなく、従来の実メモリ系のオペレーティングシステムの領域が、すべてユーザ領域と共存あるいは、核、表類、実質的スーパーバイザルーチンだけを、スーパーバイザ領域で行ない、JOBコントローラ、エディタ、コンパイラなどを、ユーザ領域に存在させていた方式から、一歩進んだものと考えられる。すなわち、スーパーバイザ領域には、スーパーバイザに関するソフトウェアがすべて存在し、ユーザ領域は、ユーザが自由に使用可能となる。

最後に本方式は、東京農工大学工学部数理情報工学科プロジェクトPIEで議論されたものであり、共同研究者各位の寄与に対し心から感謝の意を表する。

参考文献

- (1) "MC68000 USER'S MANUAL", Second edition, MOTOROLA, 1980.1
- (2) 武部、他、"16ビットマイクロコンピュータのアドレス空間に関する問題と解決策"、情報処理学会第23回全国大会、pp.949-950、1981.10
- (3) 岩崎、他、"MC68000のシミュレータの開発と、PL/IとPASCALの比較"、情報処理学会第23回全国大会、pp.951-952、1981.10